The LoopPoint methodology in the gem5 Simulator

Zhantong Qiu
Outline

● Introduction to SimPoint and LoopPoint Methodology

● LoopPoint methodology in the gem5 Simulator

● Other use cases of the LoopPoint methodology

● Summary and future work
What is sampling?
What is the SimPoint methodology?
The execution of the program is broken down into a number of intervals.
Basic Block Vector Matrix
It uses the k-mean clustering algorithm to cluster the intervals into k clusters. Each cluster has one representative interval.
By performing detailed simulations on the representative interval of the clusters and applying weights to their IPC, we can predict IPC of simulating the whole program.

\[ \frac{3}{10} \times 0B + \frac{5}{10} \times 3B + \frac{2}{10} \times 8B \]
Examples of using the SimPoint methodology can be found under the gem5 directory:
Limitation of SimPoint

- Unit of Work
- Execution Point Marker
Limitation of SimPoint

- Unit of Work
- Execution Point Marker

LoopPoint

- Using the number of loops iteration as the unit of work
- Using the PC Count pair as the execution point marker
The LoopPoint methodology:
The LoopPoint methodology:

<table>
<thead>
<tr>
<th>Region</th>
<th>PC</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>200</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
\[
\text{total runtime} = \sum_{i=\text{rep}_1}^{\text{rep}_N} \text{runtime}_i \times \text{multiplier}_i
\]
How to perform LoopPoint in gem5?

<table>
<thead>
<tr>
<th>Region</th>
<th>PC</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>200</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
The LoopPoint JSON file

region id
  |__ detailed interval
  |    |__ start
  |    |__ pc
  |    |__ global
  |    |__ regional
  |__ end
  |__ pc
  |__ global
  |__ regional
|__ warmup # optional to region
|    |__ start
|    |__ pc
|    |__ global
|__ multiplier

regional count

warmup start

detailed interval start

detailed interval end

global count

global count
The LoopPoint JSON file

```
region id
  |__ detailed interval
  |    |__ start
  |    |    |__ pc
  |    |    |__ global
  |    |__ regional
  |__ end
  |    |__ pc
  |    |__ global
  |__ regional
  |__ warmup # optional to region
  |    |__ start
  |    |    |__ pc
  |    |    |__ global
  |__ multiplier
```
The LoopPoint JSON file

```
region id
| ___ detailed interval
|   | ___ start
|   |   | ___ pc
|   |   | ___ global
|   |   | ___ regional
|   | ___ end
|   | ___ pc
|   | ___ global
|   | ___ regional
| ___ warmup # optional to region
|   | ___ start
|   |   | ___ pc
|   |   | ___ global
| ___ multiplier

"1": {
    "detailed interval": {
        "start": {
            "pc": #4221392,
            "global": 211076617,
            "regional": 15326617
        },
        "end": {
            "pc": #4221392,
            "global": 219060252,
            "regional": 23310252
        }
    },
    "multiplier": 4.0,
    "warmup": {
        "start": {
            "pc": #4221056,
            "count": 23520614
        }
    }
},

"2": {
    "detailed interval": {
        "start": {
            "pc": #4206672,
            "global": 1
        },
        "end": {
            "pc": #4221392,
            "global": 6861604,
            "regional": 6861604
        }
    },
    "multiplier": 1.0
}
```
How to take checkpoints for LoopPoint sampling?

1. LoopPoint Data File
2. LoopPoint Loader
3. Setup workload
4. Simulation loop
5. LoopPoint Save Checkpoint Generator
6. Checkpoints

Exits at start of region
The process is similar for running the region.
Examples using the LoopPoint methodology can be found under the gem5 directory:
Other use case

- The LoopPoint module uses the **PcCountTracker** to track the PC count pairs.
- **PcCountTracker** will raise an exit event at all the inputted PC Count pairs.
- For example, if we want to stop at **0x40086** when it has been executed **100** times:

```python
from m5.params import PcCountPair
from m5.objects import PcCountTrackerManager
from gem5.components.processors.simple_processor import SimpleProcessor

processor = SimpleProcessor(
    cpu_type=CPUPerfs.TIMING,
    isa=ISA.X86,
    num_cores=9,
)

target_pccountpairs = [PcCountPair(0x40086, 100)]

manager = PcCountTrackerManager()
manager.targets = target_pccountpairs
for core in processor.get_cores():
    core.add_pc_tracker_probe(target_pccountpairs, manager)
```
Other use case

- The LoopPoint module uses the `PcCountTracker` to track the PC count pairs.
- `PcCountTracker` will raise an exit event at all the inputted PC Count pairs.
- For example, if we want to stop at `0x40086` when it has been executed 100 times:

```python
from m5.params import PcCountPair
from m5.objects import PcCountTrackerManager
from gem5.components.processors.simple_processor import SimpleProcessor

processor = SimpleProcessor(
    cpu_type=CPUTypes.TIMING,
    isa=ISA.X86,
    num_cores=9,
)

target_pccountpairs = [PcCountPair(0x40086, 100)]

manager = PcCountTrackerManager()
manager.targets = target_pccountpairs
for core in processor.get_cores():
    core.add_pc_tracker_probe(target_pccountpairs, manager)
```
Other use case

- The LoopPoint module uses the **PcCountTracker** to track the PC count pairs.
- **PcCountTracker** will raise an exit event at all the inputted PC Count pairs.
- For example, if we want to stop at **0x40086** when it has been executed **100** times:

```python
from m5.params import PcCountPair
from m5.objects import PcCountTrackerManager
from gem5.components.processors.simple_processor import SimpleProcessor

processor = SimpleProcessor(
    cpu_type=CPUTypes.TIMING,
    isa=ISA.X86,
    num_cores=9,
)

target_pccountpairs = [PcCountPair(0x40086, 100)]

manager = PcCountTrackerManager()
manager.targets = target_pccountpairs

for core in processor.get_cores():
    core.add_pc_tracker_probe(target_pccountpairs, manager)
```
Summary
Future works

- LoopPoint Analysis in the gem5 simulator is on the way
  - More information about its current status can be found: https://github.com/darchr/gem5/tree/looppointAnalysis

- Combine regions runs and improve statistic process